Design of experiments in Very High Dimension by Lattice rule

Jean-Marc AZAÏS

Université de Toulouse
ENBIS, St Etienne, july 2th 2009

A billion of points in $\mathbb{R}^{1000}$

In many situations in Design of Experiments, we have to consider very high dimensional spaces and to choose a large number of points inside this space.

This is the case for example

- in the FAST method in sensibility analysis
- In Surface Response when we want to approximate a code by kriging.

A billion of points in $\mathbb{R}^{1000}$

In many situations in Design of Experiments, we have to consider very high dimensional spaces and to choose a large number of points inside this space.

This is the case for example

- in the FAST method in sensibility analysis
- In Surface Response when we want to approximate a code by kriging.

Let d the dimension of the space we consider. It will be supposed so large that no grid method is possible ($d > 100$).

This is a case if we consider a code with many entries like meteorogical variables
heath, humidity, pressure at various levels.

The sizes considered imply that classical design like orthogonal arrays or Latin Hypercube Sampling (LHS) are not tractable.

Very few methods seem available.

We will present a method constructed for cubature of integral that achieve this goal and even much more (up to $n = 10^8$).

Cubature of integrals

Let h a function defined on $[0,1]^d$ we want to compute

$$I := \int_{[0,1]^d} h(\mathbf{t}) d\mathbf{t}$$

using a finite sum

$$\widehat{I} = 1/n \sum_{i=1}^{n} h(\mathbf{t}_i) d\mathbf{t}_i$$

We define the error

$$E(h, n, \overrightarrow{\mathbf{t}}) := I - \widehat{I}$$

$\overrightarrow{\mathbf{t}}$ is the sequence of vectors $(\mathbf{t}_1, ..., \mathbf{t}_n)$

the magical tool : RKHS

We will assume that the function h is in somme class of functions $\mathcal{H}$
Possibilities are

- tensiorial product of periodic functions in a Koborov space (functions with a certain decay of Fourier coefficients)
- Soblolev space
- All these spaces are Reproducing Kernel Hilbert Spaces : RKHS

the magical tool : RKHS

We will assume that the function h is in somme class of functions $\mathcal{H}$
Possibilities are

- tensiorial product of periodic functions in a Koborov space (functions with a certain decay of Fourier coefficients)
- Soblolev space
- All these spaces are Reproducing Kernel Hilbert Spaces : RKHS

the magical tool : RKHS

We will assume that the function h is in somme class of functions $\mathcal{H}$
Possibilities are

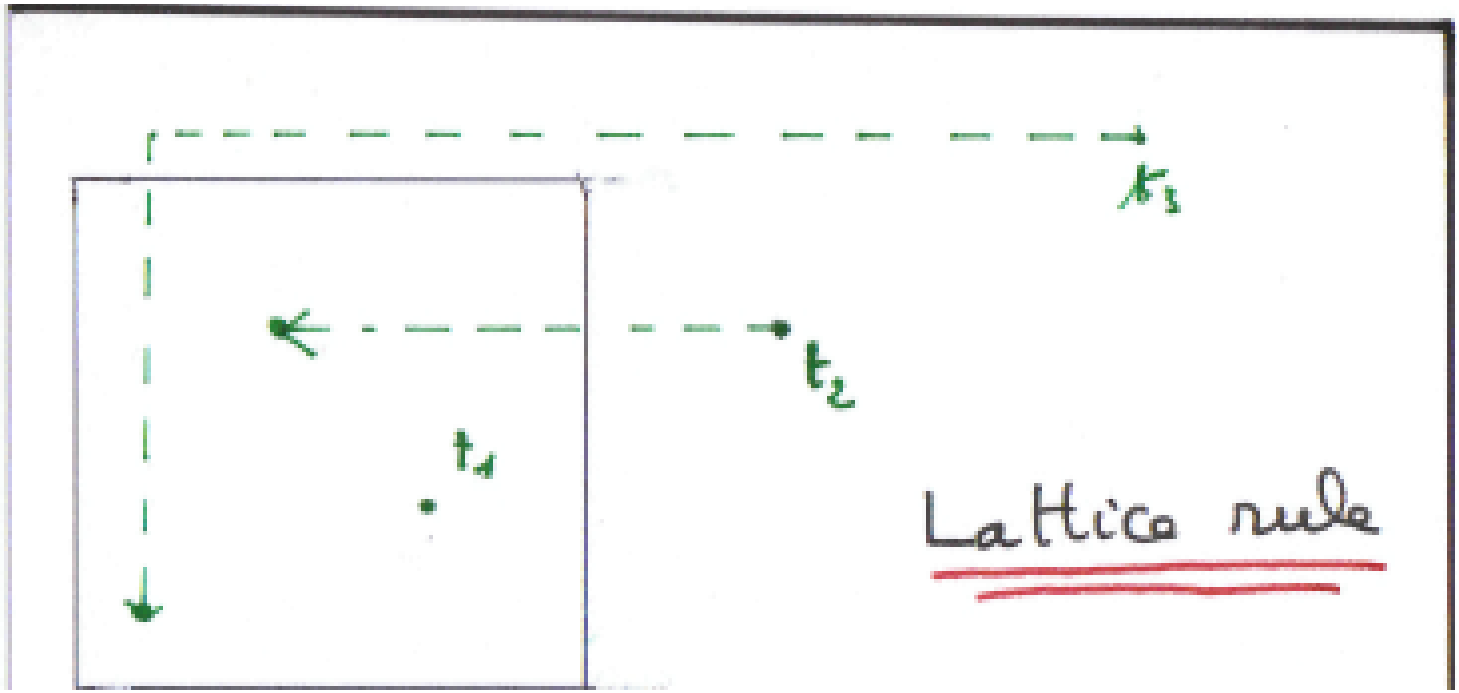
- tensiorial product of periodic functions in a Koborov space (functions with a certain decay of Fourier coefficients)
- Soblolev space
- All these spaces are Reproducing Kernel Hilbert Spaces : RKHS

Lattice rule

To simplify computations we suppose that the sequence is constructed using a **Lattice rule**.
Let $\mathbf{Z}$ be a "nice integer vector" in $\mathbb{N}^d$, the rule consist of choosing

$$\mathbf{t}_i = \left\{ \frac{i.\mathbf{z}}{n} \right\},$$

where the notation $\{\}$ means that we have taken the fractional part componentwise. n is chosen prime.


t_3
t_2
t_1
Lattice rule

Thus, now, the cubature error is a function of $d, n, \mathbf{Z}, f$:

$$E(d, n, \mathbf{Z}, h).$$

and if we take for $\mathcal{H}$ the unit ball of the preceding Hilbert spaces we can define

- the worst error $E^+((d, n, \mathbf{Z}, \mathcal{H})$
- the mean error $\overline{E}((d, n, \mathbf{Z}, \mathcal{H})$

Thus, now, the cubature error is a function of $d, n, \mathbf{Z}, f$:

$$E(d, n, \mathbf{Z}, h).$$

and if we take for $\mathcal{H}$ the unit ball of the preceding Hilbert spaces we can define

- the worst error $E^+((d, n, \mathbf{Z}, \mathcal{H})$
- the mean error $\overline{E}((d, n, \mathbf{Z}, \mathcal{H})$

Theorem

(Nuyens and Cools, 2006) consider the first space $\mathcal{H}$ (Koborov) and the worst error.
Then the worst error E^+ can be calculated explicitly using the kernel.
And a fast algorithm gives the minimax starting vector $\mathbf{Z}$
Numerical results show that the convergence is roughly $\mathcal{O}(n^{-1})$.

Same kind of result for mean error in Sobolev spaces.

How can we use the result

Nobody really believes that the function is really in the Koborov space. It is just a By-pass to obtain a nice starting vector.

The obtained result is difficult to apprehend intuitively : vectors are strange.

It is very difficult to represent graphically because we are in very large dimension.

But it works perfectly well.

The obtained sequence has low discrepancy.

How can we use the result

Nobody really believes that the function is really in the Koborov space. It is just a By-pass to obtain a nice starting vector.

The obtained result is difficult to apprehend intuitively : vectors are strange.

It is very difficult to represent graphically because we are in very large dimension.

But it works perfectly well.

The obtained sequence has low discrepancy.

How can we use the result

Nobody really believes that the function is really in the Koborov space. It is just a By-pass to obtain a nice starting vector.

The obtained result is difficult to apprehend intuitively : vectors are strange.

It is very difficult to represent graphically because we are in very large dimension.

But it works perfectly well.

The obtained sequence has low discrepancy.

How can we use the result

Nobody really believes that the function is really in the Koborov space. It is just a By-pass to obtain a nice starting vector.

The obtained result is difficult to apprehend intuitively : vectors are strange.

It is very difficult to represent graphically because we are in very large dimension.

But it works perfectly well.

The obtained sequence has low discrepancy.

How can we use the result

Nobody really believes that the function is really in the Koborov space. It is just a By-pass to obtain a nice starting vector.

The obtained result is difficult to apprehend intuitively : vectors are strange.

It is very difficult to represent graphically because we are in very large dimension.

But it works perfectly well.

The obtained sequence has low discrepancy.

How can we use the result

Nobody really believes that the function is really in the Koborov space. It is just a By-pass to obtain a nice starting vector.

The obtained result is difficult to apprehend intuitively : vectors are strange.

It is very difficult to represent graphically because we are in very large dimension.

But it works perfectly well.

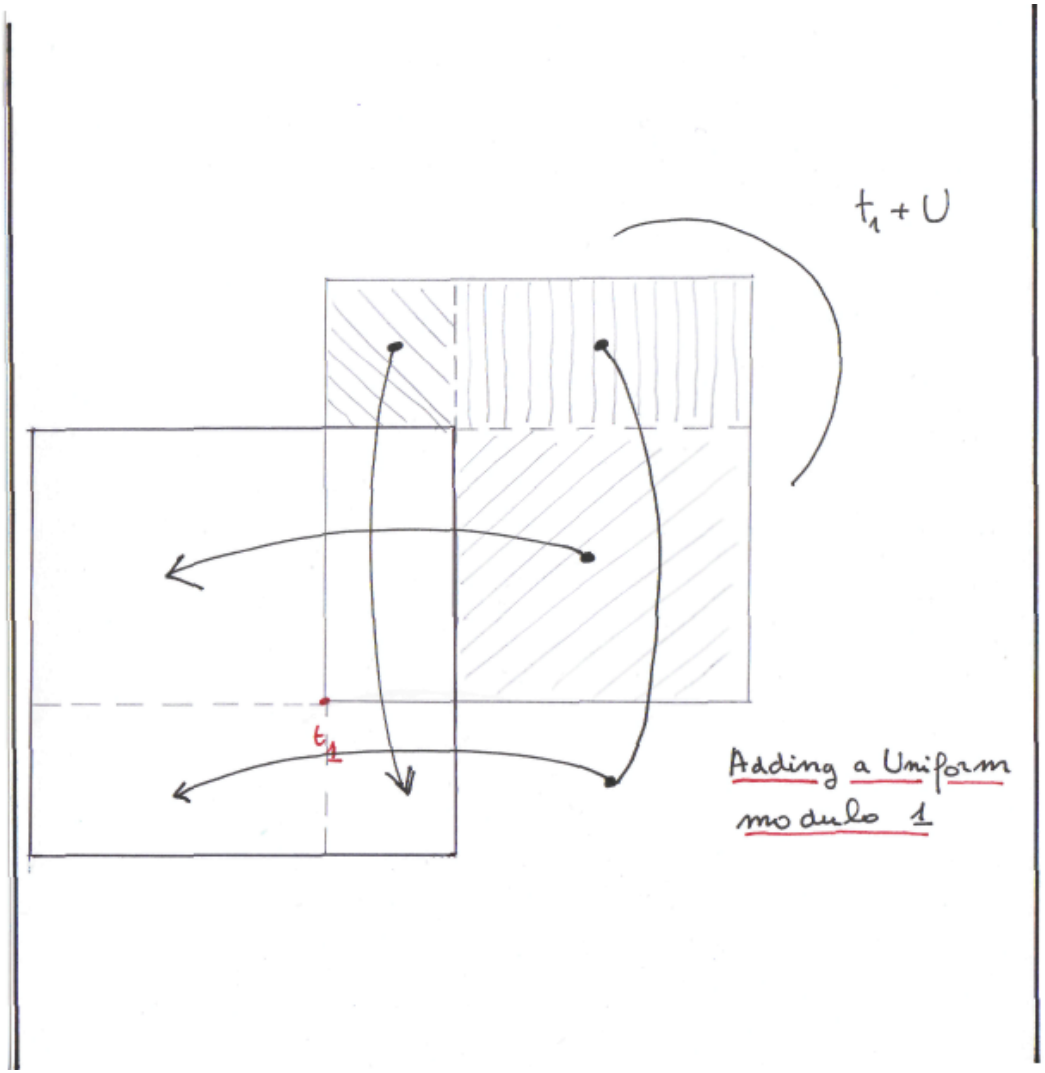
The obtained sequence has low discrepancy.

MCQMC : an extra randomisation to robustify

Let $(\mathbf{t}_1, ..., \mathbf{t}_n)$ be the lattice sequence, the way of estimating the integral can be turn to be random but exactly unbiased by setting

$$\widehat{I} = 1/n \sum_{i=1}^{n} h\big(\{\mathbf{t}_i + U\}\big)$$

where U is uniform on $[0,1]^d$.


$t_1 + U$
t_1
Adding a Uniform
modulo 1

By a meta theorem $\widehat{I}$ has small variance.

So we can make N independent replications of this calculation and construct Student-type confidence intervals. It is correct whatever the properties of the function h are.
N must be chosen small : in practical 12.

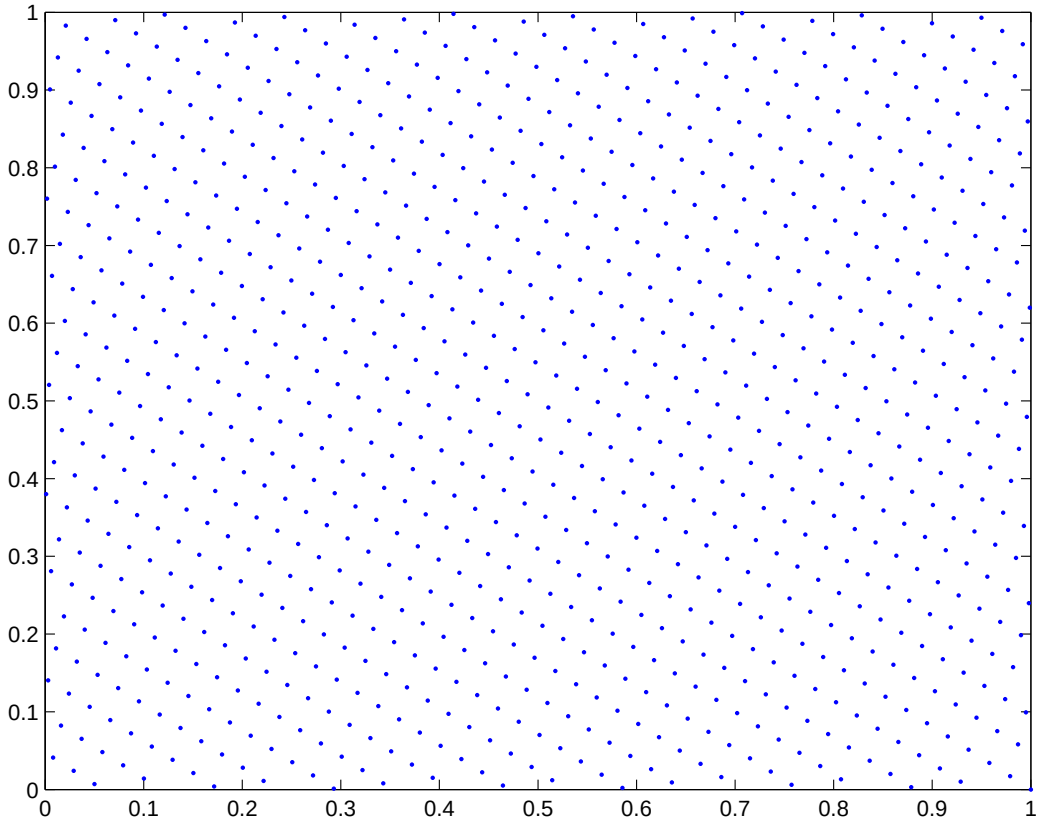
Conclusion : At the cost of a small loss in speed ($\sqrt{12}$) we have a reliable estimation of error.

An example

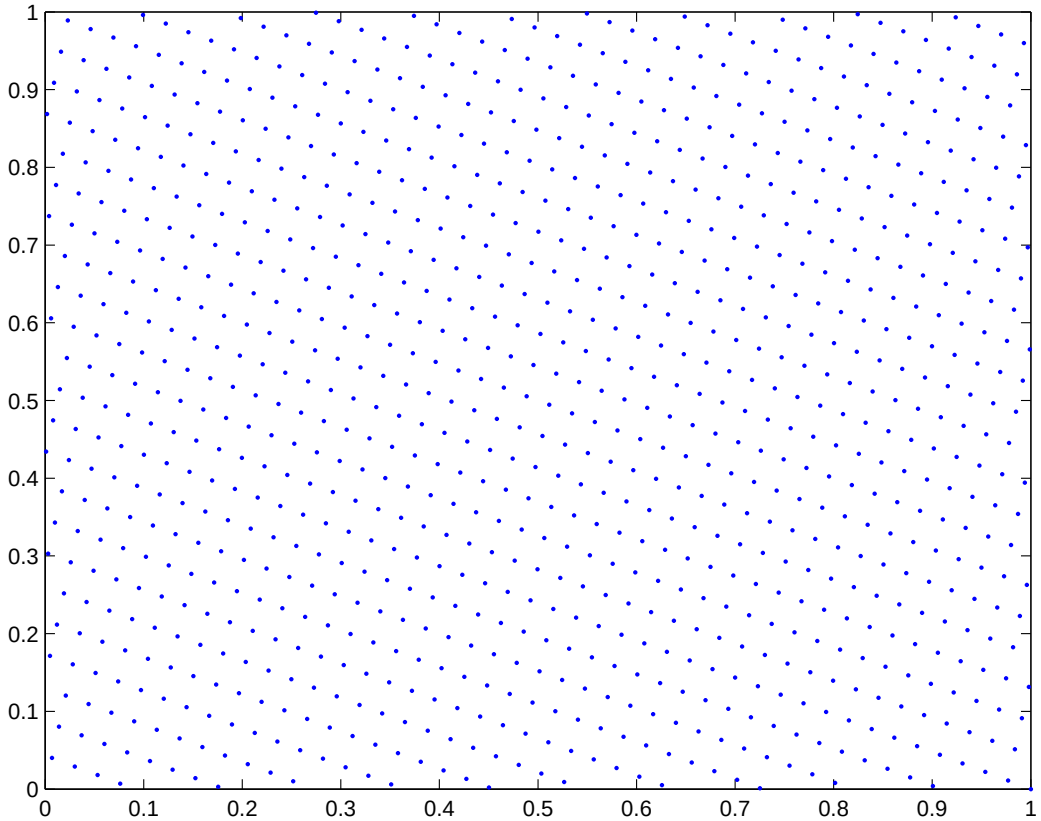
We look for (n=1000) points in $[0,1]^5$ $d=5$
In fact we choose $n=997$ which is prime and we find

$$Z=[1\ 379\ 433\ 177\ 73]$$

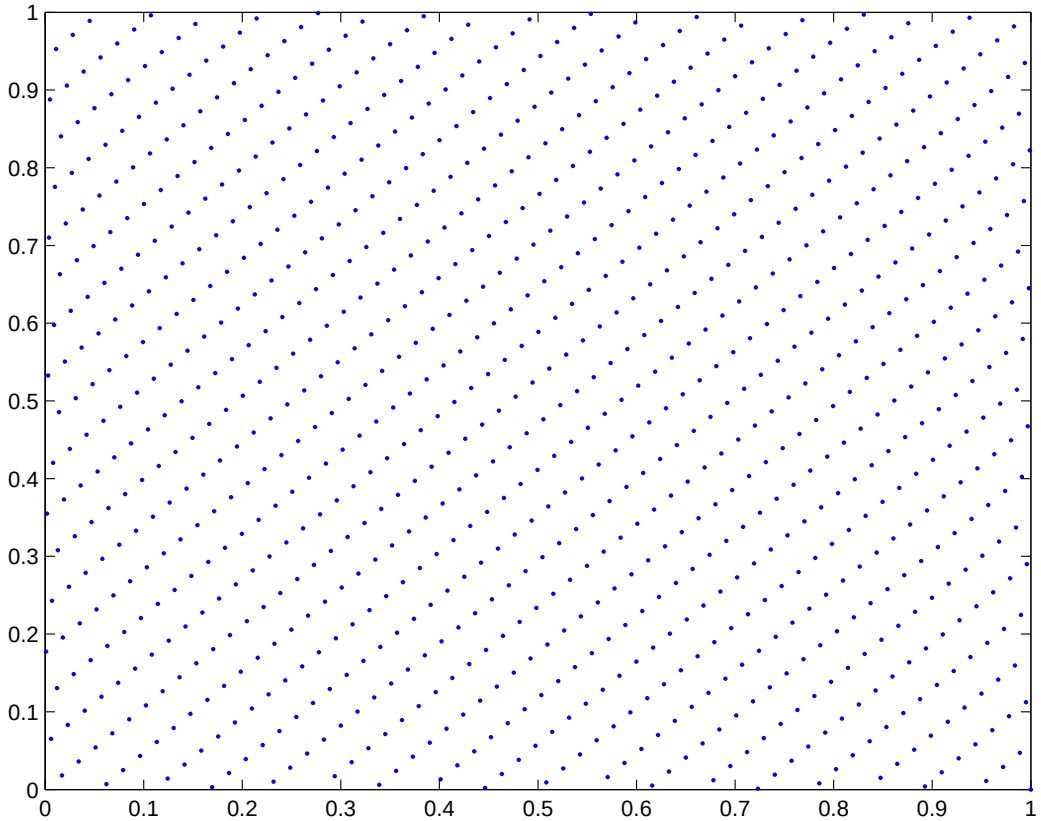
Even here graphical representation is difficult but we will try .



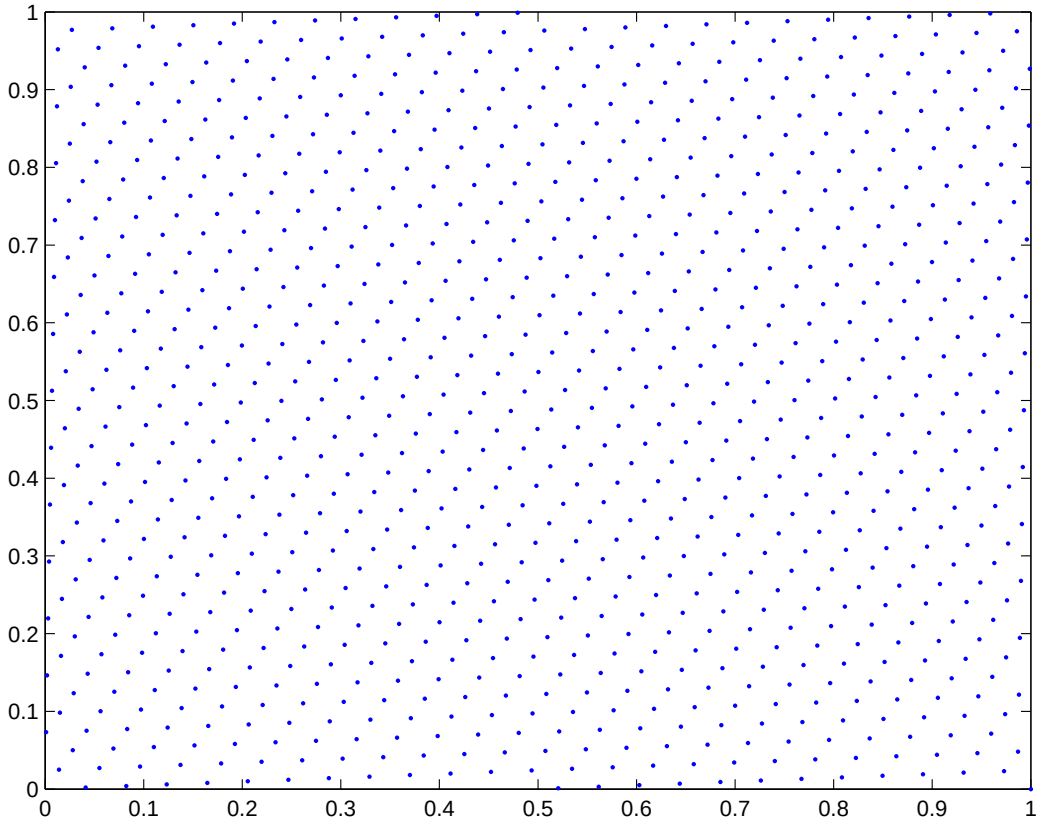
1 2



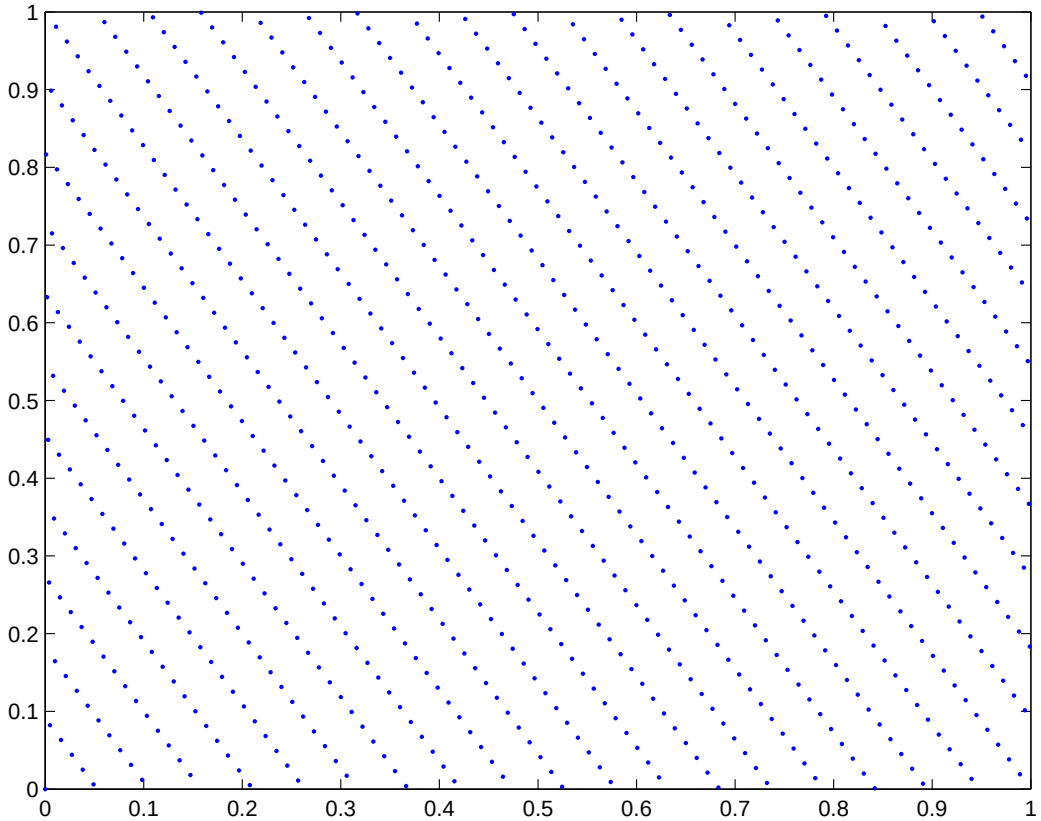
1 3



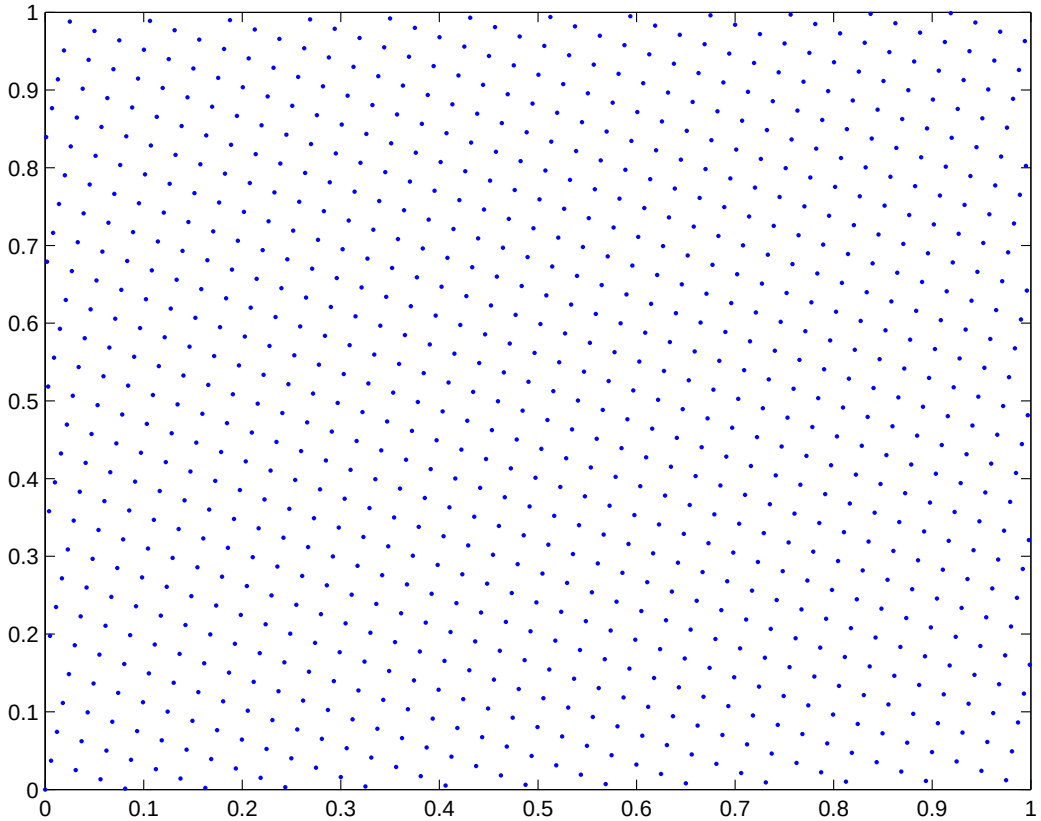
1 4



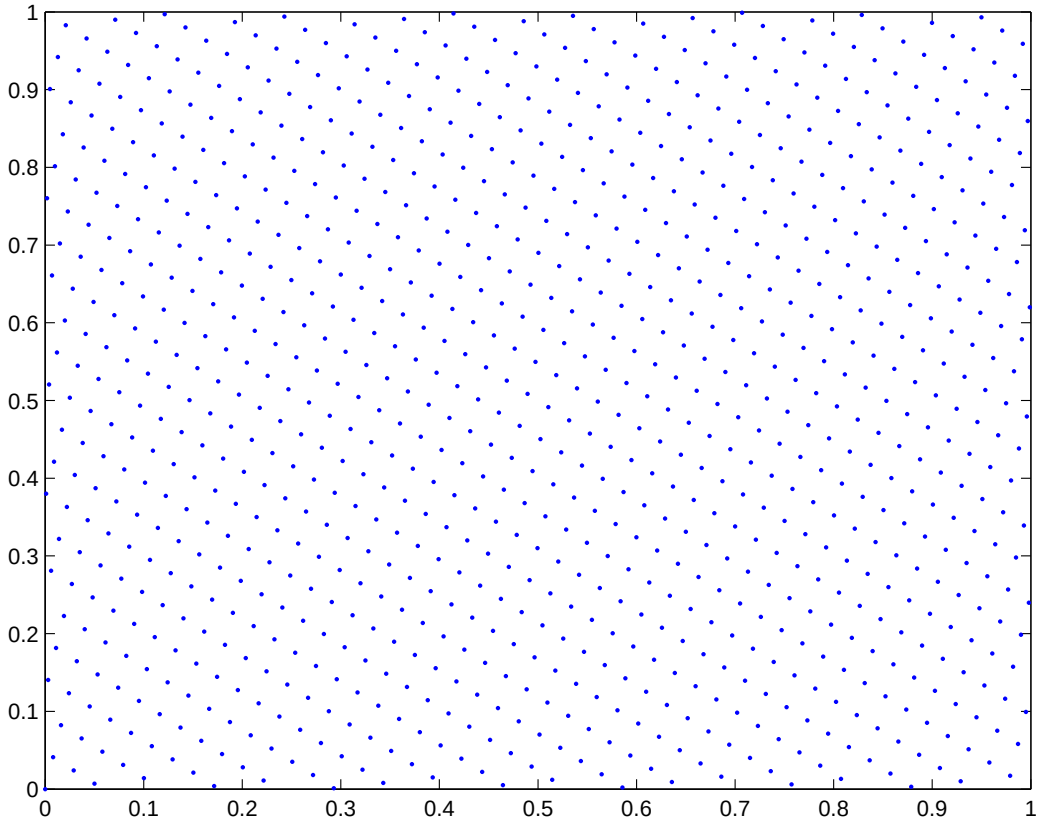
1 5



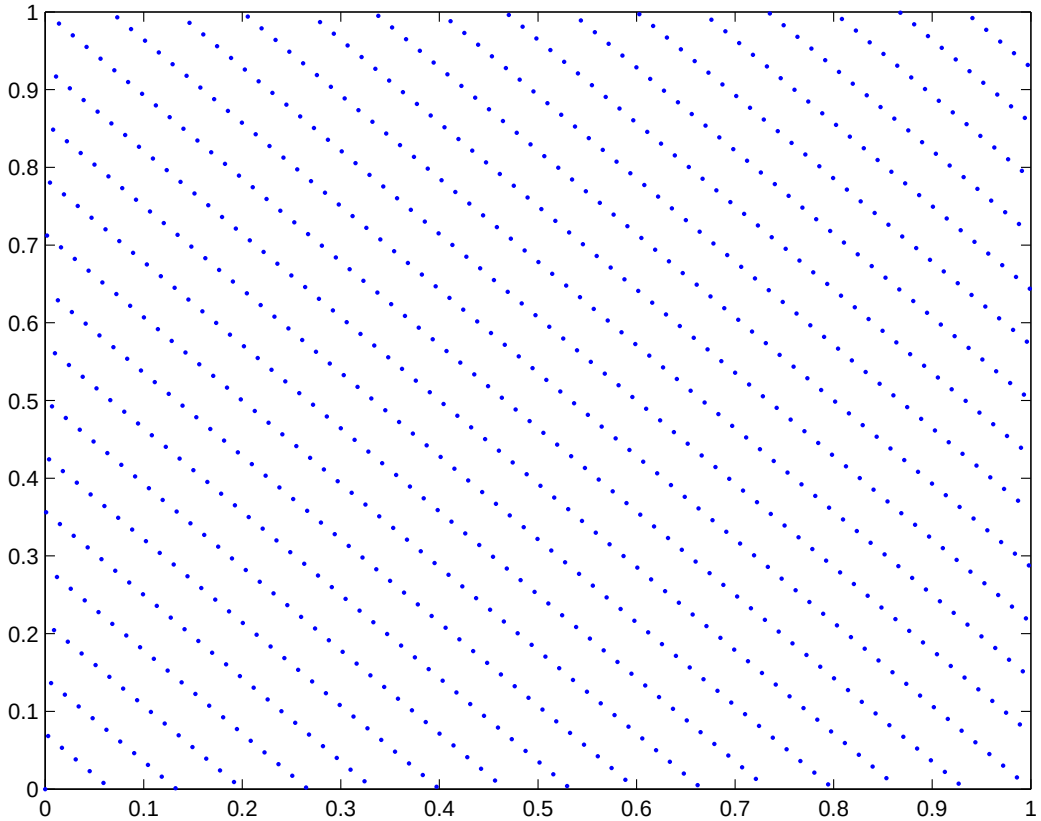
2 3



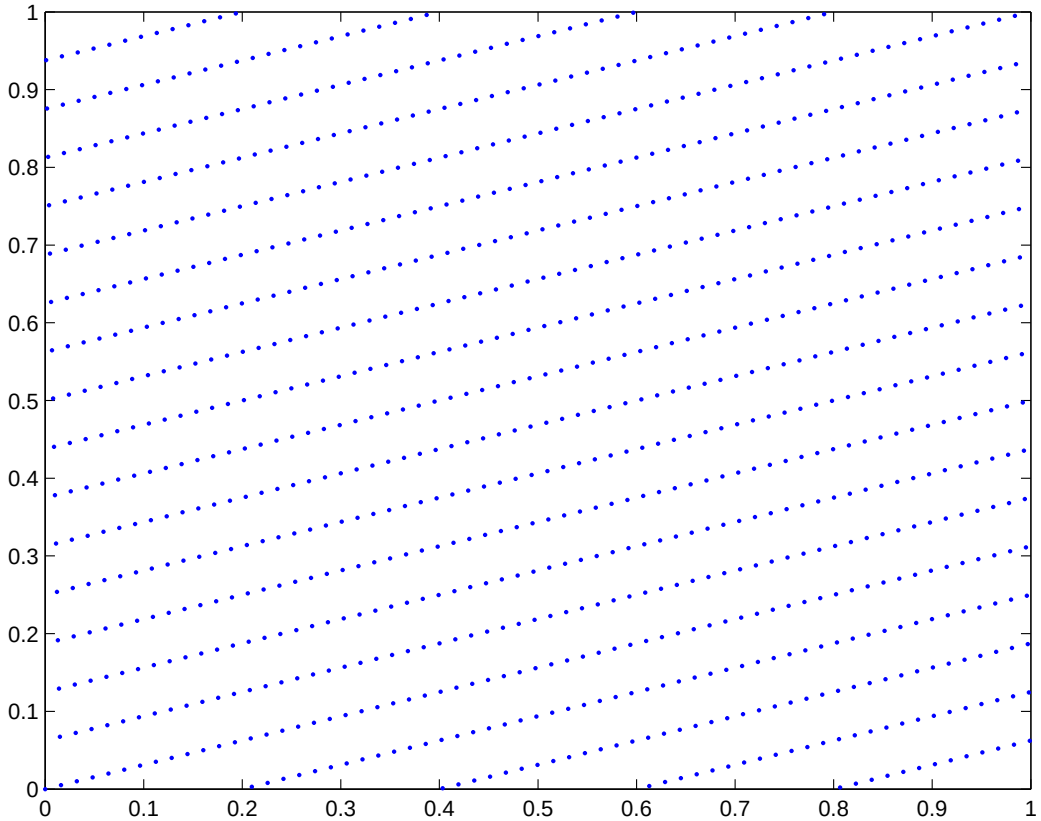
2 4



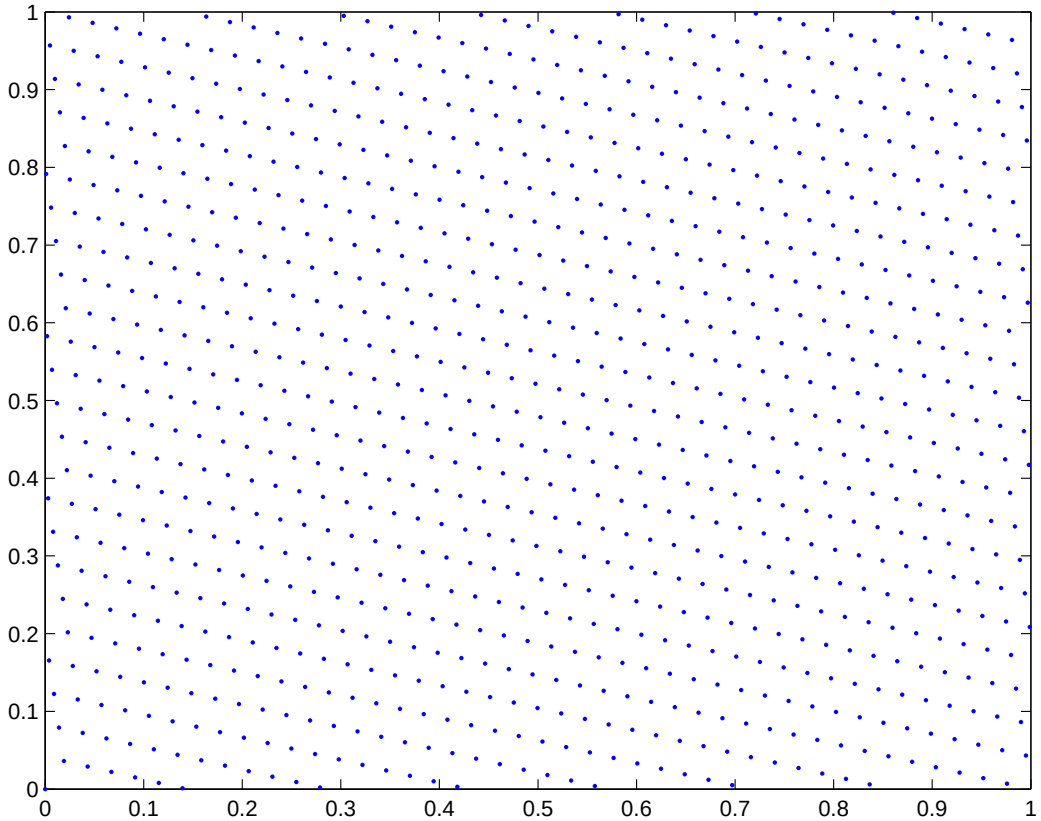
2 5



3 4



3 5



4 5

A matlab program

A program **FASTRANK** is available in the paper

”Fast Component-by-Component Construction, a Reprise for Different Kernels”, D. Nuyens and R. Cools. In H. Niederreiter and D. Talay, editors, Monte-Carlo and Quasi-Monte Carlo Methods 2004, Springer-Verlag, 2006, 371-385.

```
function [z, e2] = fastrank1(n, s_max, omega, gamma, beta)
if  isprime(n), error(n must be prime) ; end ;
z = zeros(s_max, 1) ;
e2 = zeros(s_max, 1) ;
m = (n-1)/2 ; % assume the ! function symmetric around 1/2
q = ones(m, 1) ; % permuted product vector q0 (without zero index)
q0 = 1 ; % zero index of permuted product vector : q(0)
E2 = zeros(m, 1) ; % the vector E2 in the text
cumbeta = cumprod(beta) ;
g = generator(n) ; % generator g for 1, 2, . . . , n - 1
perm = zeros(m, 1) ; % permutation formed by positive powers of g
perm(1) = 1 ; for j=1 :m-1, perm(j+1) = mod(perm(j)*g, n) ; end ;
perm = min(n - perm, perm) ; % map everything back to [1, n/2)
psi = feval(omega, perm/n) ; % the vector 0
psi0 = feval(omega, 0) ; % zero index : (0)
fft_psi = fft(psi) ;
for s = 1 :s_max
% step 2a : circulant matrix-vector multiplication
E2 = ifft(fft_psi .* fft(q)) ;
E2 = real(E2) ; % Matlab uses complex ffts : remove this noise
% step 2b : choose ws and zs which give minimal value
min_E2, w = min(E2) ; %pickindexofminimalvalue
if s == 1, w = 1 ; noise = abs(E2(1) - min_E2) , end;
z(s) = perm(w) ;
% extra : we want to know the exact value of the worst-case error
e2(s) = -cumbeta(s) + ( beta(s) * (q0 + 2*sum(q)) + ...
gamma(s) * (psi0*q0 + 2*min_E2)) / n;
% step 2c : update q
q = (beta(s) + gamma(s) * psi([w :-1 :1 m :-1 :w+1])) .* q ;
q0 = (beta(s) + gamma(s) * psi0) * q0 ;

fprintf(s=s, z(s), w, e2(s), sqrt(e2(s))) ; end ;
```

An integration example

A centered Gaussian vector of size 114 with variance matrix Σ and we want to compute the integral

$$I := \int_{-u}^{u} \cdots \int_{u}^{u} \varphi_{\Sigma}(\mathbf{x}) d\mathbf{x} \tag{1}$$

then the syntax using the public GENZ programs QSIMVNV is
a = -u*[ones(1,n)] ; b = u* [ones(1,n)] ;
[p e] = qsimvnv(40000, sigma, a, b) ; disp([p e])

Answer is : 0.5000 0.0118

Elapsed time is : 6.642098 seconds.

THANK-YOU

References

Azaïs Genz (2009),Computation of the distribution of the maximum of stationary Gaussian sequences and processes. W. paper

Allan Genz web site
http ://www.math.wsu.edu/faculty/genz/homepage

Mercadier, C. (2005). MAGP tooolbox,
http ://math.univ-lyon1.fr/ mercadier/

Nuyens, D., and Cools, R. (2006), Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces, *Math. Comp* **75**, pp. 903–920.